Amendments to the Specification

Please amend the following sections/paragraphs as indicated.

Please amend the paragraph starting on page 2, line 7, as follows:

A1

This application is related to copending U.S. Patent Application ~~Serial~~ Number ~~(Docket: MIPS:0102.00US)~~ 09/702,115, filed on October 30, 2000, entitled *Translation Lookaside Buffer for Selection of ISA Mode*, by common inventors, and having the same assignee as this application.

Please amend the paragraph starting on page 4, line 1, as follows:

A2

Because a CPU can be easily programmed to perform a wide variety of functions within a system design, within just a few years the number of CPUs and application programs in the marketplace increased exponentially. In parallel with these events, technological advances in the integrated circuit design and fabrication arts began to release a steady stream of more powerful and complex CPU designs. And as these more powerful and complex CPU designs were exploited, a number of modification and upgrade mistakes were made as a result of recoding existing application programs. So, hardware and software designers were required to focus on preserving and reusing a substantial amount of code that had already been developed and tested for use with particular CPU designs. Consequently, as newer CPUs were introduced, in addition to implementing a whole "new" set of instructions, the CPUs retained the capability to execute applications that were coded with "old" instructions. Typically, this ability to execute multiple instruction sets was bounded by a particular

A2

manufacturer's line of products. For example, Digital Equipment Corporation produced a VAX11 CPU that supported newer VAX11 instructions and older PDP11 instructions.

Please amend the paragraph starting on page 8, line 6, as follows:

A3

Years ago however, ~~Larson~~ Larsen, in U.S. Patent number 5,115,500, proposed an approach for enabling a CPU to switch ISA modes during the execution of a multiple-ISA application program that did not require the insertion of a mode switch instruction into the flow of a transferring component. ~~Larson~~ Larsen associated a program instruction's address in the CPU's address space with one of several ISA modes. In essence, ~~Larson~~ Larsen used the upper bits of the program instruction's address to indicate its ISA mode. Hence, all instructions corresponding to a specific ISA mode were stored in one or more memory segments that corresponded to that specific ISA mode. Although ~~Larson's~~ Larsen's technique addressed the issue of inserting mode switch instructions into an application program, his technique for using the upper bits of a fetched instruction's address as an indication of the instruction's ISA mode is restrictive because it requires that the CPU's address space be partitioned into fixed and equal-sized segments. And fixed, equal-sized segments do not represent the distribution of components according to different ISA modes within a multiple-ISA application program. ~~Larson's~~ Larsen's technique for switching ISA modes is inflexible and memory inefficient.

Please amend the paragraph starting on page 13, line 9, as follows:

A4

A further aspect of the present invention contemplates a method in a CPU for selecting a particular ISA mode during execution of an application program, where the

A4

application program has program instructions according to a plurality of instruction set architectures. The method includes partitioning an address space of the CPU into [[a]] address ranges, the address ranges being designated by contents of a boundary register file; mapping each of the address ranges to each of a plurality of ISA modes; and selecting the particular ISA mode for processing of the program instruction according to mapping.

Please amend the paragraph starting on page 28, line 12, as follows:

A5

According to the mode switch technique employed by CPU 330, an instruction 313, SETPSW MODE2, is first executed that directs the second CPU 320 to set a mode bit 335 within the program status word 334, thus signaling the CPU 330 to switch to ISA mode 2. An ISA 2 jump instruction 313, JMP Y, follows in the sequence that directs the CPU to transfer program control to address Y. The use of a bit 335 or bits of a program status word 334 to accomplish ISA mode switches is described by Jaggar in U.S. Patent number 5,568,646 and U.S. Patent number 5,740,461. Accordingly, during execution of component A 311, ISA 1 instructions 312 are fetched by the CPU 330 and a multi-ISA decoder 332 monitors the state of the mode bit 335 to determine which ISA decoding rules to apply for a current instruction [[324]]. When the instruction 313 is executed that modifies the mode bit 335 in the program status word 334, the multi-ISA decoder 332 detects the state of the bit 335 and begins decoding following instructions according to ISA mode 2. Hence, to execute a multiple-ISA application program according to this second technique, each time that program control is transferred to a component 315 that is encoded with instructions from an ISA that is different from the ISA of the transferring component 311, an instruction to set the mode bit 335 of the program status word 334 must be inserted into the instruction stream of the

A5

transferring component's instruction flow and the jump instruction that actually causes flow to be transferred must be encoded according to the ISA mode of the transferred component 315. One skilled in the art will appreciate that it would not be recommended to place the mode bit instruction 313 as the first instruction in the transferred program component 315 flow because the mode bit instruction 313 must be encoded according to the ISA mode of the transferring component 311, and in an application program comprising several ISA modes, the transferred component 315 could be called by components encoded in more than one ISA.

Please amend the paragraph starting on page 32, line 10, as follows:

A6

~~Larson~~ Larsen, in U.S. Patent number 5,115,500, advocated an approach for providing independent program components in a multiple-ISA application program by using the uppermost bits of a program instruction's address as means for signaling the ISA mode of the program instruction. In the specific embodiment described by ~~Larson~~ Larsen, the upper three address bits were used to determine one of two (or more) ISA decoding modes. Program components encoded according to, say, ISA 1 mode, were to be stored in a first one of eight memory segments, program components encoded according to ISA 2 mode were stored in the remaining segments (in accordance with one embodiment).

Please amend the paragraph starting on page 32, line 22, as follows:

A7

Although the technique described by ~~Larson~~ Larsen is desirable from the standpoint that program components are effectively decoupled from all other referenced program components, ~~Larson's~~ Larsen's approach is inflexible because it requires that a CPU's address space be partitioned into fixed and equal-sized segments. Practically speaking, the

distribution of instructions according to each of the ISA modes in a multi-ISA program is not uniform in any sense of the word. In fact, this distribution varies from program to program as a function of the specific requirements that are implemented and based upon the particular processor upon which the programs are executed. ~~Larson's~~ Larsen's equal-sized segment technique is disadvantageous because it does not allow memory space to be partitioned according to the specific needs of a multi-ISA application program.

Please amend the paragraph starting on page 34, line 1, as follows:

Referring to FIGURE 4, a block diagram 400 is presented illustrating a portion of a multiple-ISA processor 450 according to the present invention having a boundary address register file 460 for selection of ISA modes. The boundary address register file 460 comprises a plurality of boundary address registers 461, each containing an address boundary, BDY2-BDYN. The address boundaries, in one embodiment of the present invention, are addresses within the address space of the CPU 450 that mark lower address bounds of ISA mode address ranges. Each of the address ranges is mapped to one of a number of ISA modes that are implemented by the CPU 450. In an alternative embodiment, the addresses denote upper address bounds of the address ranges. The block diagram 400 also depicts a memory 410 having locations that span the address space of the CPU 450. Within the memory 410 are stored program instructions 412, 416, 414, 418, 419 corresponding to N different instruction set architectures. For illustrative purposes, two components, component A 411 comprised of ISA 1 instructions 412 and component B 415 comprised of ISA 2 instructions 416, are specifically stored within the memory 410 to distinguish encoding of these components 411, 415 according to the present invention from

A8

like components 311, 315 described above with reference to FIGURE 3. In addition, the block diagram 400 features program instructions corresponding to ISA 3 414, ISA N-1 418, and ISA N 419 stored in their respective address ranges in memory 410.

Please amend the paragraph starting on page 39, line 13, as follows:

A9

The write back stage 550 writes the results generated in the execute stage 530 or contents of data memory retrieved by the data stage 540 into prescribed registers in the general purpose register file. Hence, program instructions are fetched from memory 560 by the fetch stage logic 510 and synchronously proceed through subsequent CPU stages 520-550 in a fashion very much like an assembly line. Accordingly, the present invention does not require that any additional "switch" or "veneer" instructions be inserted into the pipeline flow in order to explicitly direct the CPU 500 to switch ISA modes because a given program instruction's ISA mode is implicitly carried in its corresponding address. This is advantageous from [[a]] an execution speed perspective because the insertion of additional instructions into the flow of the pipeline bogs down the execution of an application ~~program~~ program.

Please amend the paragraph starting on page 44, line 3, as follows:

A10

At block 708, the CPU according to the present invention fetches a next instruction of the application program from the memory into which [[is]] it has been loaded. Along with the next instruction, an address of the next instruction, ADDR, is fetched. Flow then proceeds to block 710.

Please amend the paragraph starting on page 47, line 5, as follows:

A11

In addition, the present invention has been particularly characterized in terms of a CPU or microprocessor. In particular, one embodiment of the present invention described with reference to FIGURE 5 portrays application ~~its application~~ within a 5-stage pipelined CPU 500. These specific embodiments and characterizations are presented herein as representative embodiments for the present invention, however, such description should by no means restrict application of the concept of basing ISA decoding mode for the processing of program instructions upon prescribed and variable-sized address ranges. On the contrary, the present invention can be embodied within a multi-ISA graphics processor, a multi-ISA digital signal processor, as well as less commonly known components to include multi-ISA communications processors, multi-ISA video processors, multi-ISA memory controllers, and multi-ISA micro controllers.